AR-010-394

Supporting a Component-Based Software Engineering Approach for the Development of Takari Products and Future Command and Control Systems

R.J. Vernik, M.P. Phillips & S.F. Landherr

DSTO-TR-0596

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Supporting a Component-Based Software Engineering Approach for the Development of Takari Products and Future Command and Control Systems

R. J. Vernik, M. P. Phillips and S. F. Landherr

Information Technology Division
Electronics and Surveillance Research Laboratory

DSTO-TR-0596

ABSTRACT

This report discusses issues related to the methods and tools required to support the development and evolution of future generations of Command Control Communications Intelligence and Information Warfare (C3I/IW) systems. It focuses on concepts and approaches defined as part of the Takari programme and considers issues related to tool support for the development of Takari Capability and Technology Demonstrators (CTDs) such as the proposed Experimental C3I Technology Environment (EXC3ITE). A Component-Based Software Engineering (CBSE) lifecycle, which includes both Domain Engineering and Product Engineering, is proposed as a possible development strategy. A tool categorisation framework is defined to help define tool requirements. An initial survey of CBSE tools provides examples of currently available tools which could be considered for each of the tool categories.

RELEASE LIMITATION

Approved for public release

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury South Australia 5108

Telephone: (08) 8259 5555
Fax: (08) 8259 6567
© Commonwealth of Australia 1997
AR No. AR-010-394
December 1997

Executive Summary

Project Takari is a DSTO research and development programme which aims to provide focused, coordinated and integrated research and development support for the development of Australian Defence Force (ADF) of Command Control Communications Intelligence and Information Warfare (C3I/IW) capability. This programme focuses on providing the R&D foundations for integrated C3I/IW systems which will provide ADF commanders with relevant, reliable, and timely information for the conduct of joint and combined operations.

An important aspect of the Takari approach is the use of Capability and Technology Demonstrators (CTDs) which will provide major foci for research. Each CTD will address specific operational capabilities such as wide area surveillance and/or C3I systems attributes (e.g. interoperability and security). For example, the proposed Experimental C3I Technology Environment (EXC3ITE) is a Takari CTD which will be used to demonstrate the potential of broadband network communications and distributed computing technology to enhance the ADF's C3I capability. This CTD will provide the generic building blocks which can be utilised in several specific capability demonstrators and in many R&D activities.

One of the major R&D areas being addressed in Takari is C3I systems issues. This area focuses on issues such as acquisition and development approaches, the definition of high-level architectures, assessment of performance and effectiveness, and a broad range of other systems and software engineering concerns. The results of this work will help ADF acquire future C3I/IW capability in a more cost-effective and timely manner and provide an effective basis for developing and transitioning CTDs. This report begins to consider some of these issues.

Current trends suggest that future C3I systems will be developed using Commercial-Off-The-Shelf (COTS) hardware and software to reduce development costs and schedules. The use of distributed object technologies may provide developers of these software-intensive systems with the basis for dealing with heterogeneity, allow for the development of more flexible and evolvable solutions, and facilitate a component-based development approach. Fitzpatrick and Jackson (1997) propose how distributed object technologies might be used for EXC3ITE. They suggest an architecture based on the use of a distributed object infrastructure layer (or Object Bus) which deals with networking, communication, and platform heterogeneity. The Object Bus also serves as a component integration framework. In systems of this type, a component is considered to be an object which supports a set of specific interfaces (e.g. Domain Interfaces, Object Services) within an Object Framework (i.e. a domain specific set of objects). A component-based distributed object architecture can be implemented using various technologies, based on several competing and emerging standards such as CORBA, DCOM, JavaBeans and ActiveX. This report considers how the development of these types of systems might be supported through an appropriate use of software engineering methods and tools.

The methods and tools required for a particular project need to be considered in terms of the activities to be supported. This report provides an overview of the types of processes and activities that need to be considered for software projects. A

Component-Based Software Engineering (CBSE) lifecycle is proposed as a strategy for projects which use a component-based development approach. This lifecycle identifies a range of processes and activities that would need to be supported including domain analysis, architecture development, component selection, and component assembly. The lifecycle is used to help define categories of tools and a tool framework for CBSE. An initial survey of CBSE tools provides examples of currently available tools which could be considered for each of the tool categories.

In addition to proposing a possible approach for the engineering of component-based software systems, this report discusses issues related to tool selection and use, and identifies areas where additional research and development may be necessary if we are to effectively support the Takari initiatives and future generations of C3I/IW systems. Issues discussed include the need to define appropriate lifecycle activities, method and tool evaluation, the need for commonality and compatibility of methods and tools, organisational capability and maturity, and the cost of using and misusing tools. This report also argues that a key area of R&D that needs to be considered is project and product visibility (including aspects of system documentation and visualisation). This area is particularly important for the C3I/IW domain where development is characterised by the use of evolutionary approaches, heterogeneity, and the use of off-the-shelf components. Effective visibility is required to support review activities and the early identification of risk. Support is also needed to aid collaborative work, product evolution, and transitioning activities.

Authors

Rudi Vernik
Information Technology Division

Rudi is a Senior Research Scientist employed in Software Systems Engineering Group, Information Technology Division. He currently leads the Visualisation and Description of Component-based Systems (VIDECS) task which is investigating how computer-based visualisation techniques can be used to support the acquisition, development and maintenance of large software systems. His research focuses on software systems visualisation techniques, component-based software engineering and information logistics. Rudi has a PhD in Computer and Information Science from the University of South Australia. He also has a Bachelor of Electronics Engineering (with Distinction) and a Diploma of Communications Engineering from the Royal Melbourne Institute of Technology.

Matthew Phillips
Information Technology Division

Matthew is a researcher employed in Software Systems Engineering Group, Information Technology Division. His research interests include distributed systems, programming language design, software visualisation and object-oriented software engineering. Matthew has a Bachelor of Computer Science (with Honours) from The University of Adelaide.

Stefan Landherr
Information Technology Division

Stefan is Head of the Software Systems Engineering Group in Information Technology Division. He has a BSc (Hons.) in mathematical physics and a graduate Diploma in Computing Science. He has worked in DSTO for 26 years on the analysis, specification and development of software for combat systems and other military systems. His current research interests include software metrics, software architectures and software re-engineering.

Contents

ABBREVIATIONS

ADF	Australian Defence Force
API	Application Programming Interface
AViDeS	Advanced Visualisation and Description of Software
C3I	Command Control Communications and Information
C3I/IW	Command Control Communications and Information Warfare
CASE	Computer-Aided Software Engineering
CBSE	Component-Based Software Engineering
CM	Configuration Management
CMVC	Configuration Management and Version Control
COM	Common Object Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial-Off-The-Shelf
CTD	Capability Technology Demonstrator
DARPA	Defence Advanced Research Projects Agency
DCE	Distributed Computing Environment
DCOM	Distributed Common Object Model
DSSA	Domain-Specific Software Architecture
EXC3ITE	Experimental C3I Technology Environment
GUI	Graphical User Interface
IDE	Integrated Development Environment
IDL	Interface Definition Language
ISPR	Integrated Software Product Release
OLE	Object Linking and Embedding
OMG	Object Management Group
OMT	Object Modelling Technique
OO	Object-Oriented
OOA/D	Object-Oriented Analysis and Design
OODBMS	Object-Oriented Database Management System
OOSE	Object-Oriented Software Engineering
ORB	Object Request Broker
RAD	Rapid Application Development
RPC	Remote Procedure Call
STARS	Software Technology for Adaptable, Reliable Systems
TP	Transaction-Processing
UML	Unified Modelling Language

1. Introduction

This report discusses issues related to the methods and tools required to support the development and evolution of future generations of Command Control Communications and Information Warfare (C3I/IW) systems. It focuses on concepts and approaches defined as part of the Takari programme (Takari 1996) and considers issues related to tool support for the development of Takari Capability and Technology Demonstrators (CTDs) such as the proposed Experimental C3I Technology Environment (EXC3ITE).

This report argues that there are three main areas that need to be considered prior to selecting development methods and tools. These are:

1. the project context (eg application domain, team size and organisational complexity, capability and maturity),

2. product characteristics (eg heterogeneous, distributed, long lived, specific technologies), and

3. process activities to be supported.

Since many of these aspects are yet to be defined for the systems of interest, recommendations for specific tools cannot be provided. However, based on current knowledge of proposed approaches and some initial information on the application domain, it is possible to discuss the types of activities that may need to be supported and provide examples of currently available tools. This report also discusses issues related to tool selection and use and identifies areas where research and development may be necessary if we are to effectively support the Takari initiatives and future generations of C3I/IW systems.

Section 2 begins by providing an overview of the Takari and EXC3ITE approaches and hence helps define the project/domain context. Section 3 then considers the types of processes and activities that may form part of this approach. These processes and activities are discussed in relation to a proposed Component-Based Software Engineering (CBSE) lifecycle which includes both Domain Engineering and Product Engineering. Section 4 then proposes a framework which identifies tool categories for CBSE. This section also provides examples of currently available tools for each of the tool categories. These example tools were identified as part of an initial tool survey that was conducted as part of this work. Appendix 2 provides an overview of the survey results.

Section 4 also argues that an important category of tools that needs to be considered is one which deals with Documentation and Visualisation. This category of tools helps individuals gain visibility of project and product characteristics. This is particularly important for the development of software-intensive systems since software is an intangible and malleable product which can be difficult to visualise. This report argues that the complexity of projects which use evolutionary development approaches for the development of heterogeneous distributed systems will require enhanced visualisation support to help monitor project status, identify risks, understand products, and to support collaborative work. The Integrated Visualisation Approach (Vernik 1996) which has been developed by DSTO to provide improved visibility of software projects and products is proposed as a means of dealing with these problems. As discussed in Section 4.9, this approach uses computer-based visualisations as a means of accessing, integrating, customising and adapting underlying project information to individual needs. These visualisations can be used to describe various characteristics such as project status, configuration status, software architecture, relationships between architectural models and implementations, and component attributes. In addition, this approach could also be used to visualise the dynamic and performance aspects of executing C3I systems and constituent software. The use of this approach for the development and evolution of component-based C3I/IW systems may help reduce documentation costs and provide a basis for more timely identification of software-related problems and risks.

Section 5 provides a summary of issues that need to be considered in relation to tool support for future C3I/IW projects. Section 6 provides conclusions and recommendations.

2. Overview of Takari Approach

Project Takari is a DSTO research and development programme which aims to provide focused, coordinated and integrated research and development support for the development of Australian Defence Force (ADF) C3I/IW capability to the year 2010. This programme focuses on providing the R&D foundations for integrated C3I/IW systems which will provide ADF commanders with relevant, reliable, and timely information for the conduct of joint and combined operations. An important aspect of the Takari approach is the use of Capability and Technology Demonstrators (CTDs) which will provide major foci for research. Each CTD will address specific operational capabilities such as wide area surveillance and/or C3I systems attributes (eg interoperability and security). CTDs will deliver research results in a form which facilitates understanding of the implications for operational capability. They will provide components for "Battle Labs" through which scientists can work closely with ADF combat personnel to refine appropriate technological solutions to operational problems and to explore new technology-enabled operational concepts.

One of the major R&D areas being addressed in Takari is C3I systems issues. This area focuses on issues such as acquisition and development approaches, the definition of high-level architectures, assessment of performance and effectiveness, and a broad range of other systems and software engineering concerns. The results of this work will help ADF acquire future C3I/IW capability in a more cost-effective and timely manner and provide an effective basis for developing and transitioning CTDs. This report begins to consider some of these issues.

The proposed Experimental C3I Technology Environment (EXC3ITE) is a Takari CTD which will be used to demonstrate the potential of broadband network communications and distributed computing technology to enhance the ADF's C3I capability. This CTD will provide the generic building blocks which can be utilised in several specific capability demonstrators and in many R&D activities. EXC3ITE will also allow the study of a broad range of C3I systems and software engineering issues.

The trend is that future C3I systems will be developed using Commercial-Off-The-Shelf (COTS) hardware and software components to reduce development costs and schedules. Current trends suggest that these software-intensive systems will use distributed object technologies to deal with heterogeneity, provide support for the development of flexible and evolvable solutions, and to facilitate the integration of components. Fitzpatrick and Jackson (1997) propose how this approach might be used for EXC3ITE. They suggest an architecture based on the use of a distributed object infrastructure layer (or Object Bus) which deals with networking, communication, and platform heterogeneity. The Object Bus also serves as a component integration framework. In this type of architecture, a component is considered to be an object which supports a set of specific interfaces (eg Domain Interfaces, Object Services) within an Object Framework (ie a domain specific set of objects). A component is an encapsulated entity with a distinct immutable identity whose services can be accessed through well-defined interfaces. An important characteristic of a distributed component is that its implementation is hidden from the requesting client (Vinoski 1997). This type of architecture can be implemented using various technologies based on several competing and emerging standards (eg CORBA, DCOM). This report considers how the development of these types of systems might be supported through an appropriate use of methods and tools.

3. Developmental Processes and Activities

The activities to be supported for a particular project/product context need to be considered prior to defining the methods and tools to be used. This section provides an overview of the types of processes and activities that should be considered in general. It then focuses on the types of activities that would need to be supported if a component-based approach was used for the development of Takari CTDs and future C3I/IW systems.

3.1 Software Lifecycle Processes

Developmental processes often provide the focus of many planning activities. As such, prime consideration is often given to technical activities such as testing, design, and coding without appropriate emphasis on the other processes that are of major importance to a project, such as those which deal with acquisition and management. There is ample evidence to suggest that the technological aspects of a project have less effect on success or failure than these organisational and supporting aspects (Fenton 1991).

The international standard on "Software Lifecycle Processes" (ISO/IEC_12207-1995 1995) has been developed to provide a more complete definition of the types of processes, activities, and tasks that that may need to be defined and supported as part of a project. This standard considers a wide range of primary processes such as acquisition, supply, development, operation, and maintenance. As described in Appendix 1, the standard also considers the organisational processes such as management, training and infrastructure; and a range of supporting processes (eg documentation, configuration management, quality assurance, and review). Although this standard provides a general overview of the types of processes that need to be considered, the actual lifecycle approach for a project needs to be defined in terms of the project and product characteristics.

3.2 Component-Based Software Engineering

As discussed in Section 2, current trends suggest that the EXC3ITE CTD and future C3I/IW systems will be developed using a component-based approach and evolutionary development processes. A new discipline of Component-Based Software Engineering (CBSE) is emerging to support the development and evolution of these types of systems.

CBSE makes use of component integration technologies (eg Object Request Brokers), the increasing availability of off-the-shelf components, and object-oriented approaches for the development of timely, cost-effective, and adaptable systems (Brown 1996). Key features of this approach include:

- Definition and use of software architectures.
- Use of application frameworks to provide product infrastructure.
- Specification, selection, and evaluation of components.
- A focus on component assembly rather than code development.
- A product-line approach supported by Domain Engineering.

3.3 Domain Engineering and Product-Line Development

If a component-based approach is to be used for the development of Takari CTDs, and ultimately the next generation of C3I/IW systems, the processes and activities that form the basis of this approach need to be considered. Work conducted through the DARPA sponsored Domain-Specific Software Architecture (DSSA) project (Macala, Stuckey et al. 1996) provides some insights and ideas which could be used in the development of an appropriate approach for the domain of interest. This work focuses on reusing software assets as part of a product-line development.

The DSSA project defined a dual lifecycle approach which includes Domain Engineering and Application (or Product) Engineering. Domain Engineering captures knowledge about a specific domain and develops reusable assets that can be used by the

Product Engineering process to develop products which have similar characteristics (ie a family of products). The domain assets can include domain models, domain-specific architectures, application generators, and software modules. This approach has been used by Software Technology for Adaptable, Reliable Systems (STARS) demonstration projects in various domains such as Air Vehicle Training Systems, Space Command and Control, and Intelligence-Electronic Warfare. As discussed in Section 3.4, a review of the experiences and lessons learned from the DSSA work suggests that elements of this approach may be applicable to the development of the next generation of C3I/IW systems, particularly if a component-based approach is used.

Other organisations have also been active in the area of Domain Engineering and domain-specific reuse. For example, product-line development is a major theme of research being conducted at the U.S. Software Engineering Institute. The Software Productivity Consortium has developed the Synthesis method which uses various Domain Engineering approaches for supporting domain-specific reuse (O'Connor, Mansour et al. 1994). No known work in this area is currently being undertaken in Australia.

3.4 An Example CBSE Lifecycle

Figure 3.1 proposes a lifecycle model, based on the dual lifecycle approach, for the development and use of CTDs in Takari. A key process in the lifecycle is Domain Engineering. This is an iterative process which produces and evolves assets. These assets can be used in the development of a family of capability demonstrators and could also be used to support the development of operational C3I systems. The Domain Engineering process involves capturing knowledge about the C3I/IW domain in a set of domain models. These models provide the basis for developing domain-specific architectures. Figure 3.1 shows the software architecture as one of the key assets provided by Domain Engineering. The software architecture can be used to acquire a range of other assets such as integration frameworks and components. The acquisition of components may involve a range of activities including component specification and evaluation. Domain Engineering also provides a basis for defining common tools and approaches and for recording lessons learned.

As shown in Figure 3.1, Domain Engineering supports Product Engineering by making available a range of information and other assets for the development of a family of products (in this case Takari CTDs). Given a requirement for a particular CTD, developers can conduct a Domain-Based Analysis to determine the characteristics of the required product (as defined in the Product Specs) and the approach to be taken for Product Development (as detailed in the plans). In conducting the Domain-Based Analysis, the developers use the domain and architecture models to gain a better understanding of the problem space. They view the available components, frameworks, and lessons learned to get an appreciation of how the requirement might be met. The results of this activity may result in changes to the requirement and updates to the domain specific assets.

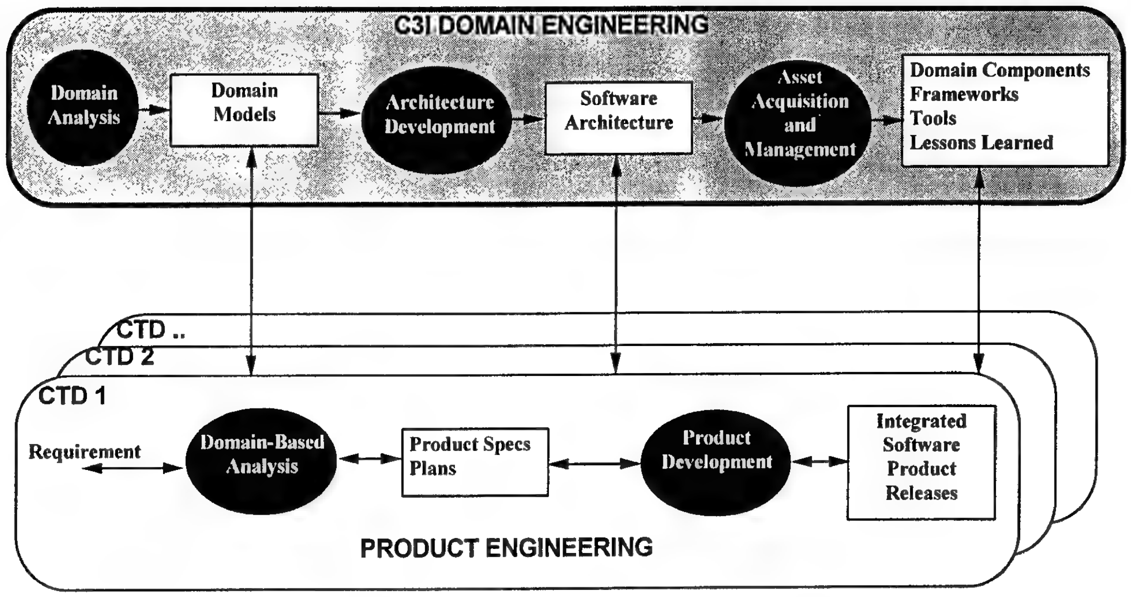

Figure 3-1 Example CBSE Lifecycle for Takari

Product Development is an iterative process which involves developing an Integrated Software Product Release (ISPR) to meet the specified requirements. This process involves elements of product design, component selection, component assembly, testing, and review. The use of domain assets plays an important role in Product Development. For example, the product design is based on the Software Architecture and Domain Models. The resulting ISPR is developed using a standard integration framework and related components. Domain assets also provide mechanisms for integrating the product into an overall architecture (eg a CTD might need to interface with other previously developed CTDs).

In addition to using domain assets, Product Engineering helps update and evolve these assets. For example, experiences gained during Product Engineering might be captured in lessons learned or through changes to domain and architecture models. Components developed during Product Development may be made available as domain assets. In some cases, a complete CTD might be considered to be a domain asset which might ultimately be integrated with other CTDs to provide a testbed environment. The integration of the Takari CBSE lifecycle needs to be considered in terms of the overall lifecycle for C3I/IW acquisition. This lifecycle could also be based on a dual lifecycle approach which draws from the Takari Domain Engineering process to provide a basis for operational systems development. A range of additional acquisition, supply, organisational, and supporting processes would need to be considered when defining this approach. This is beyond the scope of this report.

The lifecycle approach proposed in this section allows domain knowledge to be separated from a particular software product instance. Traditionally, there has been a problem in that domain knowledge has been embedded in the product (often as annotations to the source code). This information is difficult to extract and hence is often lost when the product is upgraded or redeveloped. Moreover, this approach does not effectively support reuse or evolutionary approaches. Use of a dual lifecycle approach for Takari would allow DSTO to capture and transition domain assets to Defence and industry based on the experiences gained through the development and use of CTDs. It would also help establish a culture within Defence and industry, which, together with appropriate policies, would guide how future C3I/IW systems would be acquired and maintained.

4. Tool Categories and Examples

Clearly, the approach used for the development and evolution of C3I/IW systems will have a major impact when defining tools required to support the various lifecycle activities. This section proposes a framework of tool categories which could be used to define tool needs for a Component-Based Software Engineering approach. Each tool category is discussed in relation to the activities it supports and examples of currently available tools. Further information on specific tools can be found in Appendix 2.

Figure 4-1 provides an overview of the tool categorisation framework. The framework shows that a fundamental requirement for any project is the Version Control of project information such as plans, models, components, frameworks, change data etc. Another important category of tools that supports a range of processes and activities is office automation. As discussed in Section 4.10, these tools support communication through the use of electronic mail, provide a basis for sharing information (eg through the use of World Wide Web technologies), and support a range of general tasks such as word processing. These tools can also be used as the basis for other more specific tools. For example, a database system could be used to develop a rudimentary requirements management system.

The framework includes tool categories that are required for domain engineering, product development, project management, and project support. For example, component-based product development is supported by four main areas: Modelling, Component and Interface Development, Integration and Build Management, and Infrastructure and Component Management. These are discussed in Sections 4.2 - 4.5. Where product quality is of concern, support needs to be provided for Product Verification and Validation. As discussed in Section 4.6, this area includes test tools, tools for component evaluation, and performance analysis tools. Tools may also be required to support the organisational processes. For example, Project Management tools are required to help develop plans, milestones and schedules. Process and Change Management tools help control the project.

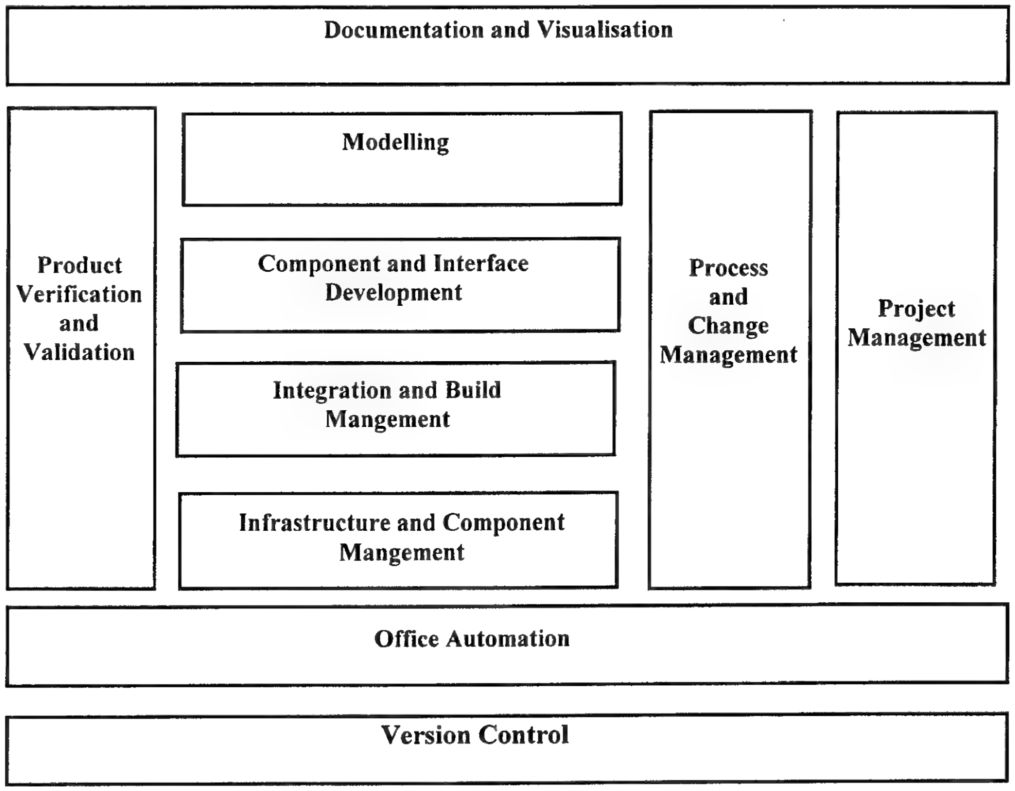
Documentation and Visualisation
Product Verification and Validation
Modelling
Component and Interface Development
Integration and Build Mangement
Infrastructure and Component Mangement
Process and Change Management
Project Management
Office Automation
Version Control

Figure 4-1 Tool Support for CBSE

An important category of tools that need to be considered is one which helps provide visibility of the entire enterprise: the processes, products, resources, and the relationships between them. These tools provide visibility of a range of project characteristics by providing needed information to those involved in a variety of activities. These tools include those that support the generation of appropriate documentation and computer-based visualisation tools that can customise and adapt information to individual needs. These tools are discussed in Section 4.9.

There is some commonality in tool requirements for the Domain Engineering and Product Engineering processes. For example, modelling tools are required to support the development of domain models and software architectures. Modelling support is also required in Product Engineering to capture specific requirements and for the development of product design models. The need for commonality and compatibility of methods and tools for the various processes is an important consideration and one which is discussed in Section 5.3.

The tool categorisation framework provides on overview of the type of support that may need to be considered for a project. Clearly, the project characteristics will determine the level of support that will actually be required. For example, a stand-alone CTD which demonstrates a particular aspect of technology, which is short lived, and which will be developed by one or two people, might simply require support for version control, rudimentary office automation, and software development (eg using a Rapid

Application Development (RAD) tool such as Borland Delphi). A more extensive CTD such as EXC3ITE which will provide the infrastructure for a range of R&D activities over a significant period of time and may involve a team of dispersed individuals, would no doubt require support for a wider range of processes including project management, domain engineering, change management, and product verification and validation.

4.1 Version Control

Version Control is a fundamental requirement for a project. Tools of this type must be able to provide control for a wide range of project information and so provide a stable repository of information. A variety of products are currently available for different computing platforms. For Windows-based development, two of the most popular version control tools are Microsoft's SourceSafe and Intersolv's PVCS. As can be seen in Table A2.1, PVCS is used as the version control system for a range of development tools. The most popular version control systems for UNIX development are SCCS and RCS. Some vendors such as Rational use these products as a basis for their overall change management systems.

When selecting a version control system, there are a range of considerations that need to be addressed. For example, there is a need to consider the range of different forms of data that can be controlled, the platforms to be supported, and the ability of the system to integrate with other tools.

4.2 Modelling

Modelling support is required for a wide range of activities. Modelling tools can help capture and generate domain models, requirements models, software architecture models, and product design models. A wide range of tools is available to support a variety of modelling approaches such as data flow, entity relationship, data dictionaries, state models, event models, object-oriented models, decision trees, and scenarios. In terms of Object-Oriented modelling, the Booch method, Jacobson's Object Oriented Software Engineering (OOSE), Rumbaugh's Object Modeling Technique (OMT) have accounted for a large proportion of the market sector. These methods have been consolidated into a common approach called the Unified Modelling Language (UML). UML has been submitted to the Object Management Group (OMG) for adoption as a standard visual modelling language for object systems. A variety of modelling tools are available to support these methods. Currently, the most popular is Rational Rose which is offered by several vendors as a third-party add in to their development environments (eg see DEC Forte). Microsoft has also licensed this technology to provide modelling support for the Microsoft Visual Studio developers toolkit. There are many other modelling methods and tools available. For example, Ascent Logic's RDD-100 system modelling approach was used extensively to support domain modelling as part of the STARS DSSA demonstrator projects.

Some modelling tools provide software development support by way of a "round trip engineering approach". For example, Rational Rose and i-Logix Rhapsody allow code for various languages to be generated from design models. Code can also be reverseengineered back into the models thereby keeping the models and implementations consistent.

4.3 Component and Interface Development

These tools provide support for software development, particularly those aspects which deal with developing components and interfaces. The CBSE approach focuses on component development/assembly rather than traditional code development. A variety of visual programming tools have recently become available to support the development of user interfaces through the use of a "drag and drop" approach to component reuse (eg Microsoft's Visual Basic, Borland Delphi,). In addition to creating and using custom controls, many tools now also provide support for the generation and use of ActiveX components. There has also been significant interest in support for Java development through the development and use of JavaBeans components. Several product offerings will support this approach including Sun's Java Workshop, Borland's Jbuilder, IBM's VisualAge, and Symantec's Visual Café. Support for the development of interface software for various middleware products (eg using CORBA IDL) also needs to considered in terms of this category of tools.

4.4 Integration and Build Management

This tool category supports application development by providing support for component integration and the generation of Integrated Software Product Releases (ISPR) or software 'builds'. Various tools have been developed to support these functions. For example, the UNIX 'make' system provides a rudimentary form of build management. Integrated environments such as Rational Apex extend these concepts to support large teams and complex product families. Application generators such as Genesis and Predator (Batory, Singhal et al. 1994) have been used to synthesise complex software systems from reusable components. These types of tools are needed to generate and manage the various product releases developed by the project. They are particularly important when teams of people are involved and an evolutionary development process is being used.

Although there are several approaches and tools available to support the generation of ISPRs, they provide support for predominantly homogenous systems. Future C3I/IW systems will be distributed heterogeneous systems built using a range of COTS components, application frameworks, and incorporate complex interfaces to legacy systems. The use of component integration technologies such as CORBA and DCOM show promise for supporting the generation of these types of systems. Some tools are beginning to emerge which support the integration of some COTS components such as databases. For example, Borland's Jbuilder will support the integration of Visigenic's Object Request Broker and a range of database systems, and Oracle has proposed an approach for integrating legacy databases into distributed object applications. However, more sophisticated tools will be required to more fully support these activities.

4.5 Infrastructure and Component Management

A component-based development approach relies on ready access to components. Support is required to manage domain components and the integration framework. During product development, there is a need to be able to quickly identify and gain access to a required component. This type of support is beginning to emerge in relation to ActiveX/COM components and the Microsoft VisualStudio toolset. Support is provided for managing available components in a component registry. Integrated support is also provided for viewing component descriptions including interface properties. This type of support will also be provided for JavaBeans and CORBA through tools such as Borland's J-Builder and Symantec's Visual Café.

4.6 Product Verification and Validation

Product Verification and Validation tools are required to help assess the quality of products. One area that needs to be considered is the area of software testing. A variety of testing tools are available and they support a range of needs. For example, Aonix StP will generate test cases from requirements and Pure Atria's ClearCoverage can provide test coverage information. Microsoft VisualTest is a tool which integrates with VisualStudio and Rational tools and is specifically designed to support a component-based approach.

Validation of models is another area that may need to be considered. For example, tools such as i-Logix Rhapsody provide animation to support the validation of Object-Oriented design models.

Other tools in this category include tools that support component evaluation (eg by way of metrics tools), performance measurement and monitoring tools, and tools such as Pure Atria's Purify which conducts memory use testing. Consideration also needs to be given to tools that support the analysis of usage data that would be obtained from instrumented applications. These types of techniques and tools play a key role in supporting research activities and can provide a basis for more effective use of systems (Vernik 1996).

4.7 Project Management

Project Management tools provide support for developing plans, schedules and milestones. Microsoft Project is a example of this type of tool. Other project management activities that may need to be supported include cost estimation/tracking and risk management. Some integrated environments such as Ascent Logic's RDD-100 provide support for project management. However, in most cases these are stand-alone tools which at best are loosely integrated into an environment.

4.8 Process and Change Management

Process and change management plays an important part in a project. It can involve a range of activities such as task allocation, defect tracking, and problem reporting. Many of the configuration management functions can be considered part of this category. Version Control provides the basic control of information items for a project. Additional functions are required to capture and record change requests, define actions to be undertaken, map actions to individual tasks, and invoke appropriate processes. Process and change management of this type is required for a wide range of project artefacts such as requirements, architecture models, domain models, and software components. Examples of tools that provide this type of support are Rational's Summit and Pure Atria's ClearCase. Other tools that might be considered in this category are tools that provide process measures and requirements management tools.

4.9 Documentation and Visualisation

Tools in this category are particularly important since they help individuals gain visibility of project and product characteristics. The information provided can be used to support a variety of activities such as project status reviews, product reviews, risk assessment, and process improvement. This type of information also supports collaborative work. It needs to be provided in a timely and cost-effective manner.

This type of support can be provided in a variety of ways. For example, several vendors provide automated documentation generation systems which consolidate information obtained from various sources (eg CASE tools) into a standard documentation format. Tools such as Ascent Logic's RDD-100 and Rational's SoDA can provide this type of support. Key problems with this type of approach are that the information provided is not tailored to a particular need, the resulting documents can be expensive to produce and use, and the information provides details at a particular point in time and so may not be an accurate representation of current status.

Several tools have reporting facilities which can be used to help gain insights into particular project or product characteristics. However, a major problem can result when information from several sources needs to be assimilated into a more complete picture. Moreover, information from specific tools cannot generally be customised or adapted to a particular need. The Advanced Visualisation and Description of Software (AViDeS) R&D task (ALO 94/081) (http://www-se.dsto.defence.gov.au/avides/) has been conducted by Information Technology Division to study these issues. One of the key outcomes of this task is the definition an Integrated Visualisation Approach which uses novel computer-based visualisation techniques to provide more effective information to those involved in tasks associated with the acquisition, development, and maintenance of software-intensive systems. This approach (as implemented in an Integrated Visualisation Environment) uses a Composite Systems Model to support the integration of information and views. The approach has been used to access and integrate information from a range of CASE tools and other sources during the development of predominantly homogenous software systems. These studies suggest that this approach may prove of benefit for the development of Takari CTDs and future C3I/IW systems where the effectiveness of system visibility may determine success or failure.

4.10 Use of Office Automation Tools

The use of office automation tools is an important consideration since they can be used to support a variety of needs. These include tools such as a word-processor, spreadsheet, database, electronic mail, and web browsing/publishing. In addition to supporting communication amongst team members and other stake holders, these tools can be used to develop other tools. For example, instead of acquiring a fully integrated process and change management tool such as Rational Summit, a small project may only require a simple defect tracking system which could be cost-effectively developed using a general database such as Microsoft Access. When selecting office automation tools for a project, consideration needs to be given to how the tools will integrate with other tools that may have been chosen.

5. Summary of Issues

A range of issues need to be considered in terms of providing support for the development of Takari CTDs and future C3I/IW systems. In addition to discussing these issues, this section identifies areas where additional research and development may be required if appropriate support is to be provided.

5.1 Defining the activities to be supported

As previously discussed, specific tools cannot be selected until there is a clear definition of the activities to be supported and the technologies to be used. Activities are defined in terms of the project characteristics and the approaches used for development. This report has provided examples of a component-based approach and a CBSE lifecycle which could be used as a basis for developing future C3I/IW systems. Work needs to be undertaken to more fully define the overall acquisition and development approach (and the associated tool categories) to provide guidance to those involved with selecting and procuring tools. Knowledge of available tools for the C3I/IW domain needs established and maintained.

Those activities that are important but are not supported effectively by available tools must also be identified. There may be a need to develop specific tools (eg using office automation tools as a basis) or generic tools such as the Integrated Visualisation Environment proposed in Section 4.9 to support these needs.

5.2 Evaluation of methods and tools

Evaluations may need to be conducted to support method and tool selection and to help gain an understanding of their limitations. For example, scalability and maturity of methods and tools can be major problems (Vernik and Turner 1992). There is a need to assess whether a particular method is suitable for the task at hand. For example, several visual modelling methods may be applicable for a particular task but which is the most appropriate? Which tool best supports the method? Evaluation may be done by applying the methods and tools in pilot projects and through laboratory experimentation. Evaluation activities can also be supplemented with information (eg lessons learned) gleaned from other projects.

There is a need to consider trade-offs when selecting tools. For example, a choice often needs to be made between tools that comprise an integrated toolset versus individual "best of breed" tools.

5.3 Need for Commonality and Compatibility

Commonality of methods and tools is an important consideration. There are a range of benefits that could be gained by selecting common methods and tools for Takari developments. For example, commonality would allow for improved communication between various stakeholders, DSTO teams, and industry. It may also provide a better basis for collaborative work. Commonality of methods and tools may also allow for more effective transfer of people between teams, reduce training costs, and provide for more effective use of technologies (eg through the establishment of user groups which would provide a basis for the sharing of knowledge and experiences and support the development of project standards). Although there are many areas where the selection of standard tools and methods might be applicable (eg modelling approaches, version control), sufficient flexibility must be allowed to support those areas where "best of breed" approaches might be applicable. Rather than dictating a standard set of methods and tools for Takari, mechanisms should be established to suggest recommended solutions based on an appropriate evaluation of options.

The compatibility of methods and tools also need to considered. For example, will a selected development tool integrate with the project's version control system? How effectively does a particular CASE tool support a selected modelling approach? Many tools have proprietary data stores and do not allow external access to underlying information. This can hinder tool and information integration and hence limit the ways in which the tool can be used.

5.4 Organisational Capability and Maturity

Prior to selecting tools, there is a need to consider capability and maturity of the organisation (Humphrey 1989) and thus its ability to effectively incorporate and use particular types of tools. If processes and activities are not clearly thought out and defined (ie the organisation is at a low maturity level in terms of software engineering), the tools may not be used or may be used inappropriately. To be effective, tools must be integrated into well-defined development processes.

5.5 Cost of Using and Misusing Tools

The purchase cost of the tool is only a proportion of the cost of transitioning a tool into an organisation. There also other costs to consider, such as the cost of training personnel in the use of the tools and the methods it supports, and the costs associated with the (re) engineering of processes to accommodate the tool. The cost of misusing tools can also be a major risk for projects. For example, individuals can become seduced by tools and lose track of the main objectives. In the case of visual modelling tools, this can result in large complex models which do not capture the essence of the problem and which can be difficult, if not impossible, to comprehend and use.

6. Conclusions and Recommendations

This report has discussed a range of issues related to the selection of tools (and their associated methods) to support the development and evolution of Takari products and future C3I/IW systems. The report argues that tools must be selected based on the project context, project characteristics, and the activities to be supported. Most of these aspects are as yet undefined for Takari. As such, recommendations for specific tools cannot be made. However, it is possible to consider possible approaches based on current software engineering practices and define particular areas where tool support may be required. This report has proposed a Component-Based Software Engineering approach as an example of an approach which may be applicable to the C3I/IW domain. Tool categories that may be required to support CBSE within this domain have been identified and discussed.

Recommendations are:

1. Emphasis needs to be directed towards defining an overall development approach for the C3I/IW domain and Takari developments. The approach needs to be considered in relation to the acquisition processes. A Component-Based Software Engineering approach which supports both Domain Engineering and Product Engineering should be considered as an option when defining the development approach and strategy. A well defined approach will provide a basis for planning and ultimately, method and tool selection.

2. A focus area should be established in ITD to capture and maintain knowledge about methods and tools applicable to the C3I/IW domain. This area may need to perform method and tool evaluations, provide guidance on selection of tools and tool use, and help support the integration of tools and project information. There is also a need to identify areas where tool support is not available and where additional R&D needs to be directed. The tools survey needs to be progressively updated with new tools. It should be extended to include information on tool costs, compatibility aspects, and specific recommendations.

3. A key area of R&D that needs to be considered is documentation and system visualisation. This area is particularly important for the C3I/IW domain where development is characterised by the use of evolutionary approaches, heterogeneity, and the use of off-the-shelf components. Effective visibility of project and products is required to support review activities and the early identification of risk. Support is also needed to aid collaborative work, product evolution, and transitioning activities. This R&D area should consider issues related to the costs and timeliness of information required for a range of project activities.

7. References

Batory, D., V. Singhal, et al. (1994). "The GenVoca Model of Software-System Generators". IEEE Software September: 89-94.

Brown, A. W., Ed. (1996). "Component-Based Software Engineering". Los Alamitos, CA, IEEE Computer Society Press.

Fenton, N. E. (1991). "Software Metrics: A Rigorous Approach", Chapman and Hall.

Fitzpatrick, M. and D. Jackson (1997). "Issues in C3I Architectures", Distributed Systems Technology Centre. Brisbane, QLD.

Humphrey, W. S. (1989). "Managing the Software Process", Addison-Wesley, New York.

ISO/IEC_12207-1995 (1995). "Information Technology - Software Life Cycle Processes", International Standards Organisation.

Macala, R. R., L. D. Stuckey, et al. (1996). "Managing Domain-Specific Product-Line Development." IEEE Software, May : 57-67.

O'Connor, J., C. Mansour, et al. (1994). "Reuse in Command and Control Systems". IEEE Software,**Sep** : 70-79.

Takari (1996). C3I/IW Research and Development Plan for Takari. http://www-sa.dsto.defence.gov.au/DSTO/divisions/takari/takunc.pdf, DSTO/ITD.

Vernik, R. J. (1996). "Visualisation and Description in Software Engineering". PhD Thesis, Computer and Information Science, University of South Australia: 232.

Vernik, R. J. and I. Turner (1992). "Techniques and Tools for Analysing Software Products". Australian Computer Journal **24**(3): 98-105.

Vinoski, S. (1997). "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments". IEEE Communications Magazine, 14(2) February : 46-55.

Appendix 1: Standard Software Lifecycle Processes

The term lifecycle process is often used in the literature. But what is a lifecycle process? ISO/IEC_12207-1995 defines a process as "a set of interrelated activities which transform inputs to outputs". This standard defines processes in terms of sets of activities and tasks which need to be performed. Tasks are well-defined work assignments and are defined as the smallest unit of work subject to management accountability (IEEE_Std_1074-1991). Humphrey (1995) extends these definitions by suggesting that a software process sets out the technical and management framework for applying methods, tools, and people to the software task.

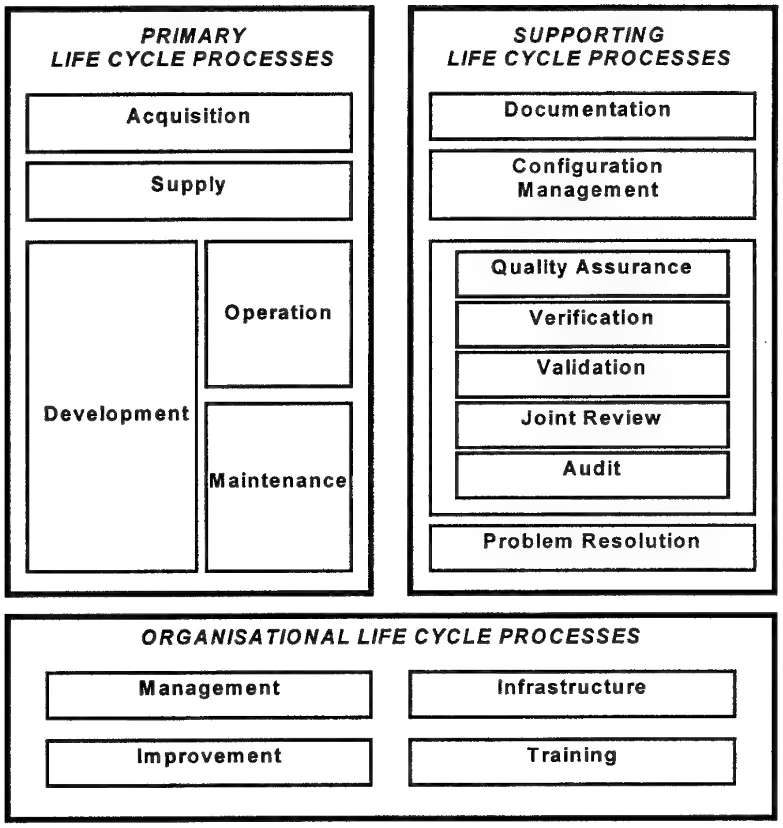

Figure A1.1 Standard Lifecycle Processes

As shown in Figure A1.1, ISO/IEC_12207-1995 defines a standard set of software lifecycle processes for software projects. These are:

- **Primary Processes.** The Primary Processes are those that serve primary parties during the software lifecycle. These primary parties are the acquirer, the supplier, the developer, the operator, and the maintainer of software products. Five processes are defined to support the activities of these parties. These are the acquisition, supply, development, operation, and maintenance processes. The development process is defined in terms of requirements analysis, design, coding, integration, testing, installation, and acceptance activities.

- **Supporting Processes.** Supporting Processes are employed and executed, as needed, by other processes. These processes are defined as the documentation, configuration management, quality assurance, verification, validation, joint review, audit, and problem resolution processes.

- **Organisational Processes.** The Organisational Processes are employed by an organisation to manage the activities being performed in other processes, provide and support the project infrastructure, improve the way in which activities are performed, and provide necessary training. The Organisational Processes are defined as the management, infrastructure, improvement, and training processes.

A project can be defined in terms of its primary, supporting and organisational processes. Various supporting and organisational processes are defined for each of the primary processes. For example, a management process may be defined for each primary process that forms part of the project context.

Appendix 2: Initial Survey of CBSE Environments and Tools

This appendix provides an overview of an initial survey that has been conducted to identify environments and tools which could be used to support the Component-Based Software Engineering Approach proposed in this report. Table A2-1 provides a summary of the available tools in relation to the tool categories defined in Section 4. Shaded regions indicate that a vendor offering supports a particular tool category. Notes are provided for each vendor offering including a brief overview of the tool and pointers to other online information.

Table A2-1 shows that some vendors provide frameworks which integrate their own tools and selected third party tools. Other vendors provide individual tools which support one or more tool categories. The table can be used in several ways. The horizontal dimension identifies the areas of support provided by a particular vendor offering. For example, it can be seen that *HP Softbench* can offer support in most tool categories. The vertical dimension can be used to identify "best of breed" tools for a specific purpose.

Table A2-1 Summary of CBSE Tools

Vendor/Tool	Version Control	Modelling	Component & Interface Development	Integration & Build Management	Infrastructure & Component Management	Product Verification & Validation	Project Management	Process & Change Management	Documentation & Visualisation
Advanced Software Technologies Graphical Designer		OOD (Booch, Rumbaugh OMT)			Design object repository				Design visualisation
Aonix StP		OOA/D (Jacobsen/ use-case/ Booch/ OMT)				Test case generation from requirements			
Ascent Logic RDD-100		Design model based on requirements Behaviour modelling	Integrates with 3rd party CASE tools			Dynamic verification facility	Allocation of requirements and tasks Cost estimation	Requirements management	Design viewing and report generation N-squared charts Model simulation
Borland Delphi, Jbuilder, C++Builder	PVCS		Component-based RAD/GUI environment		JavaBeans, ActiveX CORBA (future)				
Cayenne ObjectTeam & PepperSeed	Repository-based CM	A&D using use-case, UML, Booch (PepperSeed)	Code generation from model		CORBA IDL generation				
Continuus/ CM	VC and project CM								
DEC COHESION worX	ClearCase or PCMS		GUI development tools		ObjectBroker			ClearCase PCMS	

Table A2-1 Summary of CBSE Tools (cont)

Vendor/Tool	Version Control	Modelling	Component & Interface Development	Integration & Build Management	Infrastructure & Component Management	Product Verification & Validation	Project Management	Process & Change Management	Documentation & Visualisation
DEC Digital Application Generator			GUI development Framework for reusable components		OLE/COM Component repository				
DEC Forté	Repository-based	3rd party (Rational Rose)	GUI development Class library		CORBA, COM, DCE Component repository Class library				
DocEXPRESS									Automated documentation generation
Excel Software WinA&D		Many methods supported (OO, real-time, state diagam, etc)	Screen prototyping Code generation						Design visualisation
GEC-Marconi RDT		Requirements allocation & tracking support							Generates requirements documentation
HP SoftBench	SoftBench CM or 3rd party eg ClearCase, RCS, SCCS, PVCS	3rd party eg, StP	UIM/X GUI builder		CommonPoint framework CORBA (OrbPlus)	3rd party (eg. ClearCase, Purify, ClearCoverage)		3rd party via tools like Pure Atria ClearCase	Files, object dependencies. 3rd party documentation tools (Interleaf)
IBM	Team Connection		VisualAge products WebRunner development framework	VisualAge WorkFrame	Team Connection Repository COM, CORBA				Code browser

Table A2-1 Summary of CBSE Tools (cont)

Vendor/Tool	Version Control	Modelling	Component & Interface Development	Integration & Build Management	Infrastructure & Component Management	Product Verification & Validation	Project Management	Process & Change Management	Documentation & Visualisation
i-Logix Rhapsody		UML, state diagrams	Automatic	Automatic	Automatic	Assists validation of design model			Design model browser Animation
i-Logix StateMate		Design via state diagrams	Code generation			Validates system design Requirements traceability			StateMate Documenter
iLOG			Views GUI development Rapid prototyping		Server distributed object manager (CORBA) Broker (CORBA/ONC)				
Intersolv	PVCS Component repository CMVC		Visual and non-visual component development	Component assembly and deployment PVCS configuration builder	Distributed component repository		Project status monitoring & completion estimation	Requirements and change management Workflow & inter-communication	Requirements documentation Project status reporting
Microgold WithClass		OOA/D multiple methodologies	Code generation						Design visualisation
Microsoft Visual Studio	Visual SourceSafe	Visual Modeller	GUI builder MFC		COM/ActiveX	Visual Test			
MKS Source Integrity	MKS CM system								
Objective Spectrum Bridgepoint	ObjectStore OODBMS	OOA modelling	Code generation from model			Design model simulation & testing			Design visualisation & simulation
ObjectQuest	Object repository	OO modelling tool	GUI designer		CORBA / ActiveX	Testing tools			Design visualisation

Table A2-1 Summary of CBSE Tools (cont)

Vendor/Tool	Version Control	Modelling	Component & Interface Development	Integration & Build Management	Infrastructure & Component Management	Product Verification & Validation	Project Management	Process & Change Management	Documentation & Visualisation
Oracle	Repository	Designer OO modelling	GUI layout		OLE				Design visualisation
Platinum CCC/Harvest	Version control and project CM				Component repository		Project status & metrics data Lifecycle management		Project status
Popkin System Architect	Object repository PVCS	OO modelling (UML)	Screen painter		ActiveX				Design visualisation
PowerSoft	ObjectCycle	S-Designor	Code/object generation from design		PFC framework				Object/design browsers
Pure Atria	ClearCase					Purify PureCoverage	ClearGuide	ClearCase ClearGuide ClearDDTS	
Rational	Apex CMVC (RCS-based)	Rose (UML, Booch, etc)	Apex Rose code generation	Apex	COM/ActiveX	TestMate		Summit CMVC Requisite Pro	SoDA
Sun Java Workshop	RCS and others		Workshop IDE Java framework	Workshop project manager	JavaBeans, CORBA (NEO)				
Symantec Visual Café Pro			Forms-based RAD environment	Café project manager	JavaBeans (future), Café repository				Class hierarchy visualisation

NOTES

Advanced Software Technologies Graphical Designer

- UNIX/Windows OO design (Booch, Rumbaugh OMT), fast design layout, generates full C++, Java or Smalltalk (future) code from design.
- Reverse engineering of C++ and Java to OMT possible.
- Multi-user capabilities (version control, rollback).
- Can produce reports for use with Office tools and others.
- Object repository.
- http://www.advancedsw.com/

Aonix Software Through Pictures (StP)

- **StP**: suite of OOA/D (Jacobsen, use-case, UML, Booch) tools. Can automatically produce test cases from Booch/OMT designs and requirement specs (http://www.ide.com/Products/StP_T.html).
- **007**: merger of **StP** and **ObjectAda** is an OO development environment focusing on reliable software. Offers navigation between StP models and Ada code (http://www.ide.com/Products/007.html).

Ascent Logic RDD-100

- Suite of powerful tools for requirements driven development.
- Used by multi-disciplined teams to analyze, specify, track, verify, document and manage large, complex enterprises and development projects.
- Supports the complete system-level design and development process and interfaces directly to CASE, CAE and other component development and publishing tools.
- Supports documentation generation in either custom or standard (eg. MIL-STD 490 and DoD-STD 2167A) formats.
- Supports requirements management/allocation.
- Cost estimation.
- http://www.alc.com/productfamily.html.

Borland

- **Delphi**: component-based RAD tool for Windows (http://netserv.borland.com/delphi/). Includes visual component library, COM/ActiveX support and object repository. Intersolv PVCS version control is optional. Tight database integration.
- **C++Builder**: component-based visual RAD environment for C++, very similar to Delphi (uses same visual components and database integration technology) (http://www.borland.com/bcppbuilder). Supports ActiveX as one component model. Focus on reuse and component repository. Uses PVCS for CM.
- C++Builder's extended **Open Tools API** lets you add custom development tools directly into the C++Builder environment. Link into your favourite editor, interact with Case tools, or activate a version control management system. Automate repetitive tasks by creating your own wizards.
- **JBuilder**: C++Builder/Delphi for Java (http://www.borland.com/jbuilder/). **AppBrowser** combines the features of a project manager, a class browser, and a source code editor into a single view to make managing, navigating, and editing source code easier than in any other tool. Uses JavaBeans/ActiveX as component models. Will support CORBA 2.0 via **Visgenic** ORB (http://www.visigenic.com/).

Cayenne

- **ObjectTeam**: OO development tool (http://www.cayennesoft.com/products/solutions/objectteam_java.html). Supports modelling, multiple languages (Java, Ada, CORBA IDL), component reuse. Supports iterative development. Repository supports CM and version control and allows component storage. Works with **PowerBuilder** and **DEC Forté**.
- **TeamWork**: UNIX-based project development environment (http://www.cayennesoft.com/products/datasheets/teamwork-environment.html). Supports all phases of development.
- **PepperSeed**: UML & Booch OO design tool (http://www.cayennesoft.com/pepperseed/). Reverse engineering, code generation. Integration with Office products. Supports Microsoft's object repository standard.

Continuus/CM

- CM for Unix and Windows (http://www.continuus.com/products/productsBB.html). VC, build management, workflow management, change and problem tracking. GUI interface, graphical project views
- Supports Microsoft development tools.
- http://www.continuus.com

DEC

- **COHESIONworX**: a UNIX-based (SunOS, Digital UNIX and HP/UX) distributed multi-platform software development environment (C++, Ada) that has been developed from ASD/SEE (http://www.digital.com/info/cohesion/COHESIONworX/index.html). Includes a flexible integration framework, a graphical desktop, and a set of integrated development tools, **FUSE** a graphical programming tool and Integration Framework controls and integrates tools across a distributed UNIX network, including systems from Digital, SUN, and HP. Has two configuration management options: **ClearCase** from Pure Atria and **PCMS** from SQL Software. See BYTE article at http://www.byte.com/ART/9407/SEC10/ART6.HTM.
- **Digital Application Generator:** VB-based RAD tool for client-server/three-tier distributed applications for Windows (http://www.digital.com/info/application-generator/text/summary.html). Supports design, development and deployment of Visual Basic applications. Code can be generated from design. From one design, you can generate source code, compile, and link applications to satisfy different requirements, including language, capacity, and platform requirements. Support for ODBC and OLE. Provides application templates (presumably a development framework).
- **Forté**: OO RAD using a 4GL supporting distributed application development (http://www.americas.digital.com/forte/). Graphical user interface (GUI) designer, Object-Oriented 4GL (fourth generation language), Comprehensive class libraries, Graphical debugger (multi-task debugging), Partitioning workshop (Automatic customizable partitioning), Cross platform code generation and optimization capabilities, Automatic application distribution and testing features, Application monitoring and customizable management interfaces, Team development repository (supports distributed team development). Supports Win32 and various flavours of Unix. Middleware support for CORBA 2.0 (**ObjectBroker**), OLE, DCE and others. Integrates with Rational **Rose**.
- **ObjectBroker**: the first commercially available CORBA ORB (http://www.digital.com/info/objectbroker/). CORBA 2.0 compliant, integrates with DDE, OLE and Visual Basic. CORBA objects can be accessed via OLE Automation. Working with Microsoft to increase integration with Windows.

DocEXPRESS

- Automatic documentation in many formats (DOD, MIL, IEEE, ISO) from design model (OMT, UML, etc). Supports FrameMaker, Word, Interleaf publication.
- Support for **DocIt** from Cayenne, **Paradigm Publisher** from Platinum.
- http://www.docexpress.com/

Excel Software WinA&D/MacA&D

- Supports SE methods. Structured analysis and design (Yourdon/DeMarco and Gane/Sarson), real-time (Mealy, Moore, etc), state and flow-control, OOA/OOD methods (Shlaer/Mellor, Coad/Yourdon, Booch, OMT, Jacobson and Fusion).
- Live screen prototyping.
- Requirements traceability.
- Code generation for C, C++, Pascal, Object Pascal (Delphi), Basic and SQL.
- Unix/Mac/Win32
- http://www.excelsoftware.com/overview.html.

GEC-Marconi Requirements Database Tool (RDT)

- Captures the systems engineering design process and manages system requirements. Automates requirement tracking.
- Features: Requirements capture from external documents including MS Word; Functional allocation to subsystems, Traceability between parent and child requirements via derivation entities; Test requirement allocation to any requirement; Automated document production including requirement and test specifications, Requirement Allocation Matrices, parent-child relationships directly into MS Word; Microsoft Windows and Microsoft Access compatible; Network accessible for distributed multi-user database access; Comprehensive on-line context sensitive help.
- http://www.world.net/gecm/

HP

- **SoftBench**
  - A component-oriented C++, C, COBOL application development environment (http://www.hp.com/sesd/WhatsNew/aberdeen.html).
  - Hewlett-Packard is in the process of porting **Taligent's CommonPoint** framework-based application system to HP-UX. Essentially, CommonPoint consists of a tightly integrated suite of pre-built software components written in C++. SoftBench can be used to modify these components -- or frameworks -- resulting in customized applications. HP believes that the more object-adept users of SoftBench will soon want to use CommonPoint to build collaborative, desktop applications that can run on a wide range of hardware platforms.
  - HP provides Object-Oriented Distributed Computing Environment (OODCE), which can be used in conjunction with SoftBench to create applications that run on top of the OSF's Distributed Computing Environment (DCE). Also, a **Sub-Process Control Daemon** (SPCD) is available for SoftBench, which when ported to a range of UNIX and non-UNIX platforms supplies cross-development capabilities. In the first half of 1996, HP will roll out its ORB Plus, which will let developers create CORBA-compliant C++ applications with C++ SoftBench.
  - SoftBench is able to graphically depict the structure of an application and the objects that comprise the software.

  - SoftBench is also well suited for a distributed computing model -- both from a development and deployment perspective. **SoftBench CM** enables developers to collaborate on a project, even when they are distributed across an enterprise. For deployment, SoftBench can be used to build applications that make use of the OSF's DCE, enabling various software modules to be executed on multiple hardware platforms networked together.
  - **SoftBench CM** (http://hpcc920.external.hp.com:80/sesd/products/softcm/main.html): distributed CM system for SoftBench tools.
  - No testing support, but **CodeAdvisor** (http://hpcc997.external.hp.com:80/sesd/CA/main.html) does support checking of source code for common problems.
  - Good 3rd party tool support: http://hpcc920.external.hp.com:80/sesd/3rdparty/main.html.
- **HP OrbPlus**: a C++ CORBA 2.0 ORB for HP-UX, Solaris, NT (http://www.hp.com/pressrel/apr96/02apr96h.htm). Interoperability between OLE Automation and ORB Plus. Will have 2-way mapping from COM/OLE to CORBA.

IBM

- **VisualAge**: OO RAD environment (http://www.software.ibm.com/ad/). Supports Basic, C++, Cobol, Java, Smalltalk. VisualAge for Java just entered beta testing (http://www.software.hosting.ibm.com/ad/vajava/).
- **BeanMachine**: a visual Bean authoring tool (http://www.ibm.com/java/appletauthor/).
- **WebRunner**: "Developed by Taligent, VisualAge Webrunner supports the building and testing of JavaBeans and eases the development of high-performance, client/server applications for the web." (http://www.taligent.com/)
  - Includes **JavaBeans Migration Assistant for ActiveX**, which allows ActiveX components to be moved into the JavaBeans framework.
- **TeamConnection**: "provides a unique combination of software configuration management and repository services on a high performance system. Through the use of an open object interface, TeamConnection provides for the access and extension of an object-based information model." Can work within the VisualAge toolkit. Open and extensible OO information model for tools. Suppports Win32, OS/2 and AIX. Strong CVMC support. (http://www.software.ibm.com/software/ad/teamcon/)
- **SOM 3.0** for AIX, OS/2 and Windows NT: CORBA 2.0 (with extensions) ORB (http://www.software.ibm.com/ad/somobjects/)
- **Open Class Library**: IBM's common platform C++ library for OS and GUI facilities and is now part of the VisualAge/C++ product (OS/2, AIX, Win32)
- **OpenDoc**: cross-platform component technology based on CORBA (non-IBM info at http://opendoc.macintosh.net/). Has been dropped by Apple and Oracle. Likely be absorbed into JavaBeans.

iLog

- **iLOG Views**: Platform-independent GUI development studio, including 2D graphics engine, charts, cartographic support built on a portable C++ GUI framework (http://www.ilog.fr/Products/Views/DataSheet.html). Supports OLE/ActiveX.
- **ILOG Server**: distributed object repository (implemented as a C++ framework) that integrates application and data components (http://www.ilog.fr/Products/Server/). Supports object relations, composition, communication and reuse. "enables the deployment of state-of-the-art applications such as trading, network management, traffic supervision and C3I systems". Can work with CORBA 2.0 packages such as Orbix.
- **iLOG Broker**: allows C++ application to transparently support distributed object computing. Supports Sun ONC/RPC and CORBA backbones (http://www.ilog.fr/Products/Broker/).

- **iLOG Talk**: rapid prototyping environment for C/C++, supports prototyping in Lisp + high-level object model and compiling to C/C++ (http://www.ilog.fr/Products/Talk/).

i-Logix

- **StateMate**: system design via 3 FSM model types (http://www.ilogix.com/products/cbro.htm). Provides design automation for the systems developer through modeling, analysis, and code generation capabilities. These functions enable a design team to significantly reduce system development cycle time and improve the quality of the design. Three separate modeling views enable independent capture of behaviour, functionality, and structure. Supports Ada and C. Also optionally supports documentation via **Statemate Documentor** (http://www.ilogix.com/products/docbro.htm).
    - Allows generation of GUI mockups during design phase.
    - Can simulate and verify models.
    - VC support.
    - Requirements traceability.
- **Rhapsody**: produces complete production quality code from high level designs (UML and state diagrams) and graphically animates execution behaviour (http://www.ilogix.com/products/rhapsod.htm). This allows software developers to shift their focus from coding and debugging to design and analysis while reducing the total development cycle. Design browser, design animation for debugging design model.

Intersolv

- **Allegris:** component-based OO development framework (http://www.intersolv.com/products/allegris.htm). Supports creation (via **Workshop**), storage, assembly and configuration management (via **Object Repository**) of software components. Supports creation and assembly of visual and non-visual components. Repository integrates with **PVCS**. Supports Windows, Unix and OS/2.
- **PVCS**: software change management and version control system (http://www.intersolv.com/products/scm.htm).
    - **RequisitePro**: integrates requirements management, requirements traceability and PVCS (http://www.intersolv.com/products/pvcs-reqpro.htm). "with PVCS RequisitePro, you can control expectations and 'feature creep', reduce costs and delays and improve quality. Change history and version control is offered at the individual requirement, document and project level within PVCS RequisitePro and more fully with PVCS Version Manager, ensuring that the latest requirements are known and incorporated into the development process." Supports CMM and ISO 9000 standards. Integrates with MS Word.
    - **Configuration Builder**: automates build and construction of systems and maintains audit trails.
    - **Reporter**: generates reports on project status.
    - **Tracker**: change management & manage/prioritise workloads, trend analysis for predicting completion times.
    - **Version Manager**: version control and workflow automation.

Microgold With Class

- Win32 OOA/D tool: multi-methodology OO modelling, code generation, reverse engineering for C++/Delphi/Java/Ada/Basic and ODBC databases.
- http://www.microgold.com/

Microsoft

- **Visual Studio**: integrates Java, C++, Basic, etc languages into one IDE (http://www.microsoft.com/vstudio/news/default.htm). Complete support for ActiveX/COM for all languages including Wizards for creating ActiveX components. **Visual Test**, **Visual Modeller** integrate with this basic framework. **Visual Studio 97 Enterprise Edition** includes: Visual Basic 5.0, Enterprise, Visual C++ 5.0, Enterprise, Visual FoxPro 5.0, Visual J++ 1.1, Professional Edition Professional Edition, Visual InterDev, Visual SourceSafe 5.0, Microsoft Transaction Server, Developer Edition, SQL Server 6.5, Developer Edition, MSDN Library CD-ROM. ODBC and WWW support are also main components.
- **Visual InterDev**: an ActiveX-based WWW-based IDE (http://www.microsoft.com/vinterdev/ see also PC Week http://www.pcweek.com/news/1209/09tools.html).
- **Java SDK 2.0** includes AFC foundation classes for Java.
- **Visual Test**: fast, graphical, automated testing tool for the Studio IDE. Designed to keep up with a RAD/component development approach. Licensed by Rational for use in their development environment.
- **Visual Modeller** (not yet released): UML modelling for Visual Studio (http://www.microsoft.com/vfoxpro/vfevaluate/vismodpr.htm).
- **Model integration framework**: "Microsoft Corp. today announced that 21 leading enterprise modeling vendors will support the Microsoft® Repository and have joined in an effort with Microsoft to develop the Unified Modeling Language (UML) information model. Tools from vendors implementing this functionality can interoperate through the Microsoft Repository. This interoperability enables teams of corporate developers to easily share models developed with different modeling tools, enabling higher-quality component-based application development and reuse. Slideshow (http://www.microsoft.com/repository/articles/damaover/sld001.htm).

MKS Source Integrity

- Project-oriented CM system. Runs on many platforms, supports many IDE's (Powerbuilder, Microsoft Visual Basic, Microsoft Visual C++, Borland C++, Borland Delphi, etc). Supports client/server development.
- http://www.mks.com/solution/si/2150.htm

Objective Spectrum Bridgepoint

- Bridgepoint is the definitive Shlaer/Mellor Object Oriented Analysis (OOA) and Recursive Design (RD) tool, providing total method automation from analysis to code generation. Developed by accomplished Shlaer/Mellor practitioners, this tool set is being used by leading companies seeking to automate their entire software development process. BridgePoint customers are successfully creating reuseable analysis and design components, leaping ahead of traditional code reuse programs. The result is significantly reduced time to market, increased quality, and a more predictable software development process.
- Simulation model to test design.
- Uses ObjectStore OODBMS to for repository and version management.
- http://www.informatik.th-darmstadt.de/OS_UG/mirror/partners/var/objspec.html

ObjectQuest

- Business-oriented OO RAD environment: supports UML-like object modelling (business objects in particular), C++ code generation, GUI designer and testing tools. Built on MFC on Windows platforms. Uses CORBA or TP to manage interconnection in client/server systems. Object repository for storing objects and meta data.

- Persistence toolkit allows use of an RDBMS without using SQL.
- http://www.objectquest.com/products/prodfram.htm

Oracle

- **Designer/2000**: visual system design and modelling of applications and processes (oriented towards business processes) (http://www.oracle.com/products/tools/des2k/prodov/prodov.html). Can generate code, forms and databases from design model. Support for OLE and CORBA.
    - Open repository design allows use of 3rd party repositories. Full version/access control for repositories.
- **Developer/2000**: companion to Designer/2000. Provides GUI development and database connectivity (http://www.oracle.com/products/tools/dev2k/index.html).

Platinum CCC/Harvest

- Repository-based change and configuration management solution that synchronises development activities across heterogeneous platforms during the entire application development lifecycle.
- Unix/Win32/OS/2.
- Interfaces with MS Visual tools and PowerBuilder.
- CM, reusable component management, life-cycle modelling, project status information, process metric reports.
- http://www.platinum.com/products/appdev/cccharps.htm.

Popkin System Architect

- Data modelling, OOA/D (UML and other notations) and GUI design. Supports round-trip design/implementation method (reverse engineering). Shared repository for models includes version/access control. PVCS version control for files.
- Project documentation facility available. Project management: requirements tracking, change requests, etc.
- Screen painter for GUI development
- Can link with **PowerBuilder**.
- http://www.popkin.com/prods/product.htm

PowerSoft (Sybase)

- **PowerBuilder** 5.0: OO RAD environment (http://www.powersoft.com/products/devtools/pb50/)
    - Supports OLE Client/server development for Windows, Mac, Solaris. Includes PFC, a framework for OO applications.
    - ObjectCycle project-based CM system.
    - Internet development support.
    - Object browser
- **PowerSite**: supports building WWW-based applications using other PowerSoft tools (http://www.powersoft.com/products/internet/powersite/).
- **PowerJ/Jato**: Java RAD environment (http://www.powersoft.com/products/internet/javatool/)
    - Supports JavaBeans, ActiveX component models.

    - JDBC support.
- **S-Designor**: design tools for PowerBuilder family (http://www.powersoft.com/products/design/sd_info.html). ProcessAnalyst, AppModeller, DataArchitect, MetaWorks subcategories support process/data flow modelling, application modelling with code generation (Visual Basic), database design/generation and data dictionary management.
- **Optima++:** Delphi-like C++ RAD environment.

Pure Atria

- **ClearCase**: Provides comprehensive software configuration management (SCM), including version control, workspace management, build management, and process control that scales from small project teams to the entire enterprise, available for Windows and UNIX (http://www.pureatria.com/products/clearcase/index.html).
- **ClearGuide**: software process management (http://www.pureatria.com/products/clearcase/clearguide/cg_datasheet.html). Project planning, process modelling and definition. Integrated with ClearCase.
- **ClearDDTS**: Defect tracking and change request management system (http://www.pureatria.com/products/clearddts/index.html). Tracks change requests, such as known defects and enhancement requests, throughout the product lifecycle
- **Purify**: memory use testing (http://www.pureatria.com/products/purify/index.html).
- **PureCoverage**: test coverage monitor.
- Pure Atria Recently acquired by Rational so we should see integration of products.

Rational

- Licensed **Visual Test** from Microsoft. Rational and Microsoft agreement means that all Rational and Microsoft tools should be fully integrated.
- **Objectory**: process, environment and support tools package to enable CMM level 5 OO development (http://www.rational.com/pst/products/obj_broch.html). Supports domain object models, use-case, implementation methods, testing, reverse engineering, etc.
- **SoDA**: automatic documentation system (http://www.rational.com/pst/products/rosesoda.html). Supports requirements, analysis, design and testing documentation.
- **Apex**: Development environment for C++, Java, Ada (http://www.rational.com/pst/products/apexccpp.html). Provides support for all SE processes by combining CM, code development, architecture development and control with other Rational products such as **Rose**, **TestMate** and **SoDA**.
- **Rose**: OO modelling tool (UML, Booch, business process modelling) for Visual Basic, C++, Java, Ada, SmalltalkPowerBuilder (http://www.rational.com/pst/products/rosefamily.html). Also supports CORBA 2.0 IDL.
- Requisite **Pro** (http://www.requirement.com/reqpro.htm): requirements management tool. Integrates with Microsoft Office. Integrates with PVCS. Can search for requirements within documents. **Requisite Baseline** is a version for smaller projects (http://www.requirement.com/Reqbase.htm).
- **Summit**: change management, process automation and problem tracking for Apex (http://www.rational.com/pst/products/summit_tm.html). Can use GUI (**Rose**) to help define processes. Integrates with **MS Project** and **Excel**. Comes with predefined processes.
- **TestMate**: testing framework for Ada (ie Apex/Ada) (http://www.rational.com/pst/products/testmate_ada.html). Unix support only.

Sun

- **Java Workshop**: visual development environment for Java applications (http://www.sun.com/workshop/index.html). Latest is version 2.0 beta.
- **JDK 1.1.1**: JavaBeans, new event model, drag & drop plus clipboard support, more advanced graphic capabilities, serialisation, new event model, etc.
- **Joe 2.0**: free CORBA ORB written in Java supports IIOP (http://www.sun.com/solaris/neo/joe/index.html).
- **NEO 2.0**: Solaris NEO 2.0 is Sun's distributed object environment (http://www.sun.com/solaris/neo/solaris_neo/). Connectivity with Win32 platforms and Solstice graphical distributed object management tools. "Real-time" object conversion between COM and CORBA object systems. Bi-directional interoperability among OLE, COM and NEO object systems.

Symantec Visual Café Pro

- Forms-based RAD environment for Java. Extended component library including full support for database access (JDBC), windows-like controls and WWW-enablement.
- Will support JavaBeans component model and Java specifications for persistence, drag & drop, etc.
- Supported platforms: Win32 and Macintosh.
- http://www.symantec.com/cgi-bin/menu.cgi

TakeFive Sniff

- **Sniff** (http://www.takefive.com/products.htm): IDE for Java, C++ and others. CM, browsing.

Centerline QC/Advantage

- QC/Advantage enables teams to integrate, automate and manage multitudes of tests, tools and information for testing complex network-centric distributed applications throughout the development lifecycle.
- http://www.centerline.com/products/

Supporting a Component-Based Software Engineering Approach for the Development of Takari Products and Future Command and Control Systems

R.J. Vernik, M.P. Phillips and S.F. Landherr
(DSTO-TR-0596)

DISTRIBUTION LIST

		Number of Copies
AUSTRALIA		
DEFENCE ORGANISATION		
Task sponsor:		
DGCSS		1
PMJP2030		1
S&T Program		
Chief Defence Scientist	)	
FAS Science Policy	)	1 shared copy
AS Science Corporate Management	)	
Director General Science Policy Development		1
Counsellor, Defence Science, London		Doc Control Sheet
Counsellor, Defence Science, Washington		Doc Control Sheet
Scientific Adviser to MRDC Thailand		Doc Control Sheet
Director General Scientific Advisers and Trials	)	1 shared copy
Scientific Adviser - Policy and Command	)	
Navy Scientific Adviser		1 copy of Doc Control Sheet and 1 distribution list
Scientific Adviser - Army		Doc Control Sheet and 1 distribution list
Air Force Scientific Adviser		1
Director Trials		1
Aeronautical & Maritime Research Laboratory		
Director		1
Electronics and Surveillance Research Laboratory		
Director		1
Chief Information Technology Division		1
Research Leader Command & Control and Intelligence Systems		1
Research Leader Military Computing Systems		1
Research Leader Command, Control and Communications		1
Executive Officer, Information Technology Division		Doc Control Sheet
Head, Information Architectures Group		1
Head, Information Warfare Studies Group		Doc Control Sheet
Head, Software Systems Engineering Group		Doc Control Sheet
Head, Trusted Computer Systems Group		Doc Control Sheet

Head, Advanced Computer Capabilities Group	Doc Control Sheet
Head, Computer Systems Architecture Group	Doc Control Sheet
Head, Systems Simulation and Assessment Group	Doc Control Sheet
Head, Intelligence Systems Group	1
Head, CCIS Interoperbility Lab	Doc Control Sheet
Head Command Support Systems Group	1
Head, C3I Operational Analysis Group	Doc Control Sheet
Head Information Management and Fusion Group	Doc Control Sheet
Head, Human Systems Integration Group	Doc Control Sheet
Task Manager	1
Author	3
Publications and Publicity Officer, ITD	1
DSTO Library and Archives	
Library Fishermens Bend	1
Library Maribyrnong	1
Library Salisbury	2
Australian Archives	1
Library, MOD, Pyrmont	Doc Control Sheet
Capability Development Division	
Director General Maritime Development	Doc Control Sheet
Director General Land Development	Doc Control Sheet
Director General C3I Development	Doc Control Sheet
Intelligence Program	
DGSTA Defence Intelligence Organisation	1
Corporate Support Program (libraries)	
OIC TRS Defence Regional Library, Canberra	1
Officer in Charge, Document Exchange Centre (DEC),	1
US Defence Technical Information Center,	2
UK Defence Research Information Centre,	2
Canada Defence Scientific Information Service,	1
NZ Defence Information Centre,	1
National Library of Australia,	1
Universities and Colleges	
Australian Defence Force Academy	1
Library	1
Head of Aerospace and Mechanical Engineering	1
Deakin University, Serials Section (M list)), Deakin University Library, Geelong, 3217	1
Senior Librarian, Hargrave Library, Monash University	1
Librarian, Flinders University	1

Other Organisations	
NASA (Canberra)	1
AGPS	1
State Library of South Australia	1
Parliamentary Library, South Australia	1
OUTSIDE AUSTRALIA	
Abstracting and Information Organisations	
INSPEC: Acquisitions Section Institution of Electrical Engineers	1
Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
Materials Information, Cambridge Scientific Abstracts	1
Documents Librarian, The Center for Research Libraries, US	1
Information Exchange Agreement Partners	
Acquisitions Unit, Science Reference and Information Service, UK	1
Library - Exchange Desk, National Institute of Standards and Technology, US	1
SPARES	10
Total number of copies:	**63**

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)
2. TITLE Supporting a Component-Based Software Engineering Approach for the Development of Takari Products and Future Command and Control Systems	3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION) Document (U) Title (U) Abstract (U)
4. AUTHOR(S) R. J. Vernik, M. P. Phillips and S. F. Landherr	5. CORPORATE AUTHOR Electronics and Surveillance Research Laboratory PO Box 1500 Salisbury SA 5108

6a. DSTO NUMBER DSTO-TR-0596	6b. AR NUMBER AR-010-394	6c. TYPE OF REPORT Technical Report	7. DOCUMENT DATE December 1997

8. FILE NUMBER N9505/13/109	9. TASK NUMBER N/A	10. TASK SPONSOR DGCSS PMJP2030	11. NO. OF PAGES 46	12. NO. OF REFERENCES 12

13. DOWNGRADING/DELIMITING INSTRUCTIONS N/A	14. RELEASE AUTHORITY Chief, Information Technology Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No limitations

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS
Command control communications and intelligence
Information Warfare
Project Takari
Defense projects (Australia)
Australian Defence Force

19. ABSTRACT

This report discusses issues related to the methods and tools required to support the development and evolution of future generations of Command Control Communications Intelligence and Information Warfare (C3I/IW) systems. It focuses on concepts and approaches defined as part of the Takari programme and considers issues related to tool support for the development of Takari Capability and Technology Demonstrators (CTDs) such as the proposed Experimental C3I Technology Environment (EXC3ITE). A Component-Based Software Engineering (CBSE) lifecycle, which includes both Domain Engineering and Product Engineering, is proposed as a possible development strategy. A tool categorisation framework is defined to help define tool requirements. An initial survey of CBSE tools provides examples of currently available tools which could be considered for each of the tool categories